Homework 5: Pacman

Course: CS 221 Spring 2019
Name: Bryan Yaggi

Problem 1: Minimax

(a) Before you code up Pac-Man as a minimax agent, notice that instead of just one adversary, Pac-Man could have multiple ghosts as adversaries. So we will extend the minimax algorithm from class (which had only one min stage for a single adversary) to the more general case of multiple adversaries. In particular, *your minimax tree will have multiple min layers (one for each ghost) for every max layer.*

Specifically, consider the limited depth tree minimax search with evaluation functions taught in class. Suppose there are $n+1$ agents on the board, $a_0, \ldots, a_n$, where a_0 is Pac-Man and the rest are ghosts. Pac-Man acts as a max agent, and the ghosts act as min agents. A single depth consists of all $n+1$ agents making a move, so depth 2 search will involve Pac-Man and each ghost moving two times. In other words, a depth of 2 corresponds to a height of $2(n+1)$ in the minimax game tree.

Write the recurrence for $V_{minmax}(s,d)$ in math. You should express your answer in terms of the following functions: `IsEnd(s)`, which tells you if s is an end state; `Utility(s)`, the utility of a state; `Eval(s)`, an evaluation function for the state s; `Player(s)`, which returns the player whose turn it is; `Actions(s)`, which returns the possible actions; and `Succ(s,a)`, which returns the successor state resulting from taking an action at a certain state. You may use any relevant notation introduced in lecture.

$$V_{minmax}(s,d) = \begin{cases} \texttt{Utility(s)}, & \texttt{IsEnd(s)} \\ \texttt{Eval(s)}, & d = 0 \\ max_{a \in \texttt{Actions(s)}} V_{minmax}(\texttt{Succ(s,a)}, d), & \texttt{Player(s)} = a_0 \\ min_{a \in \texttt{Actions(s)}} V_{minmax}(\texttt{Succ(s,a)}, d), & \texttt{Player(s)} = a_1, \ldots, a_{n-1} \\ min_{a \in \texttt{Actions(s)}} V_{minmax}(\texttt{Succ(s,a)}, d-1), & \texttt{Player(s)} = a_n \end{cases}$$

(b) coding